北京化工大学

技术报告(论文)

题　目　深度学习在图像分类中的应用

专　　业　信息与通信工程

学　　生　李 傲 胜

班　　级　信研 2308

学　　号　2023200842

2023 年 12 月 27 日

深度学习在图像分类中的应用

摘 要

随着深度学习技术的蓬勃发展，其在图像分类领域的广泛应用为计算机视觉领域带来了革命性的变化。本文聚焦于深度学习在图像分类中的应用，以 CIFAR-10 数据集为基础，通过卷积神经网络（CNN）构建不同模型，对图像进行了分类精度评估。我还采用了 ResNet 等著名模型，对它们在 CIFAR-10 数据集上的分类性能进行了评估。为深入研究模型的特性，我们在实验中引入了不同的激活函数，并通过对比它们在分类任务中的表现，揭示了激活函数对模型性能的影响。此外，通过调整训练轮次等超参数，我们系统地研究了模型在不同训练轮次下的分类准确度。在实验中，我发现了训练的轮次并不是越多越好，可能会出现过拟合的现象，但是我通过调整模型，并增加 Dropout 层在一定程度上解决了过拟合的问题，同时我通过模型的优化调整使得我的分类精度也得到了很大的提升，最高可达到 0.7509。在不断的优化过程中，我对于深度学习在图像分类中的前景越来越有信心，我也将继续在这个方向不断深入研究。

关键词：深度学习，图像分类，CNN，ResNet

Application of deep learning in image classification

Abstract

With the rapid development of deep learning technology, its wide application in the field of image classification has brought revolutionary changes to the field of computer vision. This paper focuses on the application of deep learning in image classification. Based on CIFAR-10 data set, different models are constructed by convolutional neural network (CNN) to evaluate the classification accuracy of images. I also took well-known models such as ResNet and evaluated their classification performance on the CIFAR-10 dataset. In order to further study the characteristics of the model, we introduce different activation functions in the experiment, and reveal the effect of activation functions on the model performance by comparing their performance in the classification task. In addition, we systematically study the classification accuracy of the model under different training rounds by adjusting hyperparameters such as training rounds. In the experiment, I found that the number of training rounds is not always better, and there may be overfitting phenomenon. However, I solved the overfitting problem to some extent by adjusting the model and adding the Dropout layer. At the same time, through the optimization and adjustment of the model, my classification accuracy has also been greatly improved, up to 0.7509. In the process of continuous optimization, I am more and more confident about the prospect of deep learning in image classification, and I will continue to further study in this direction.

Key words: Deep learning, image classification, CNN, ResNet

目录

前　言

近年来，深度学习涌现在人们的视野之中，同时各种硬件运算设备的性能在飞速地提升，卷积神经网络（Convolutional Neural Network，CNN）也促进了图像分类、目标检测等计算机视觉相关领域的蓬勃发展，并已逐渐超越并取代传统的各种图像处理方法。卷积神经网络的想法源于生物的视觉机制，它避免了繁琐的人工特征提取，在通过卷积层进行纹理、形态等物体的浅层特征提取的同时还可以获取深层语义特征，并可以端到端直接输出分类结果。

除了卷积神经网络（CNN），还有许多其他类型的神经网络和深度学习架构，每种都设计用于解决不同类型的问题。比如说循环神经网络（RNN），它适用于处理序列数据，具有记忆和上下文理解能力。主要用于自然语言处理、语音识别等任务。还有生成对抗网络（GAN），它由生成器和判别器组成，用于生成逼真的数据，如图像、音频等。GAN 被广泛用于图像生成、风格迁移等领域。

这些架构的选择取决于任务的性质，数据的特征以及计算资源的限制。深度学习领域仍然在不断发展，新的架构和模型不断涌现，以更好地解决各种实际问题。

第 1 章 绪论

1.1 研究背景与意义

图像分类作为计算机视觉领域的经典问题之一，一直以来都受到广泛关注。近年来，随着深度学习技术的崛起，特别是卷积神经网络（CNN）的成功应用，图像分类在准确性和效率方面取得了显著的进展。

在深度学习兴起之前，图像分类主要依赖于手工设计的特征提取和传统的机器学习方法，如支持向量机（SVM）和随机森林。这些方法在处理复杂的图像数据时往往受到特征设计和泛化能力的限制，难以适应大规模和多类别的图像分类任务。

卷积神经网络（CNN）的出现使得图像分类的发展更加快速，卷积神经网络成为解决图像分类问题的重要工具。CNN 通过多层次的卷积和池化操作，可以自动学习图像的层次化特征表示，从而在不同层次上捕捉到图像的语义信息。这种端到端的学习方式使得模型能够更好地理解图像的结构和内容。同时 CNN 的多层次结构使得模型能够层次化地学习图像的特征。底层的卷积层能够捕捉低级别的边缘和纹理特征，而深层的卷积层则能够学习到更为抽象和语义上的特征。这种层次化的特征学习有助于提高图像分类的准确性。

同时大规模图像数据集的可用性，如 ImageNet，也大大推动了图像分类的发展。这些数据集提供了数百万张带有丰富标注的图像，为深度学习模型提供了充足的训练样本，使得模型能够学习到更具泛化性的特征表示。

预训练的深度学习模型，如 VGG、ResNet 和 EfficientNet 等，通过在大规模数据上进行预训练，可以迁移到特定的图像分类任务中。这种迁移学习的方法使得在相对较小的数据集上也能取得显著的性能提升，提高了模型的泛化能力。

随着深度学习技术的发展，图像分类研究逐渐涉及到多模态学习，即同时处理图像、文本和其他模态的信息。这拓展了图像分类任务的范畴，使得模型能够更全面地理解和处理多源信息。

深度学习在图像分类中的应用已经取得了显著的成果，不仅提高了分类准确度，而且拓展了图像分类的研究范围。未来，随着深度学习技术的不断发展，图像分类领域将持续迎来新的挑战和机遇。

1.2 CNN 在图像分类中的应用

随着数字图像的快速增长，传统的图像分类方法在处理复杂场景和大规模数据集时面临挑战。卷积神经网络（Convolutional Neural Network，CNN）在图像分类任务中取得了巨大成功，其应用涵盖了许多领域。以下是 CNN 在图像分类中的主要应用方面：

特征学习：CNN 通过卷积层和池化层实现对图像特征的学习。卷积层通过卷积核对图像进行滤波，捕捉到局部的视觉特征，而池化层则降低特征的空间分辨率，保留主要的信息。这种层次化的特征学习有助于网络理解图像的结构和内容。

层次化表示：CNN 的多层结构允许网络逐渐学习更高级、抽象的图像特征。底层的卷积层可以捕捉边缘和纹理等低级特征，而深层的卷积层则能够学习到更抽象和语义上的特征，有助于更准确地进行分类。

局部感受野：CNN 采用局部感受野的方式处理图像，即每个神经元仅关注输入图像的一小部分区域，而不是整个图像。这样的设计使得网络对于平移、旋转和缩放等变换具有一定的不变性，提高了模型的泛化能力。

参数共享：CNN 中的参数共享机制使得模型具有相对较少的参数数量，减少了训练所需的数据量，并降低了过拟合的风险。共享的卷积核能够在整个图像上检测相似的特征，提高了模型的统计效能。

全局平均池化：在 CNN 的最后一层通常采用全局平均池化，将整个特征图降维为一个标量，然后通过全连接层进行分类。这种池化方式能够更好地保留图像的全局信息，有助于网络对整体特征进行分类。

迁移学习：预训练的 CNN 模型在大规模图像数据上学到了丰富的特征表示，可以通过迁移学习的方式，将这些学到的特征用于特定任务的分类。这种方法使得在相对较小的数据集上也能取得良好的分类性能。

CNN 在图像分类任务中的应用不仅提高了分类准确度，而且在计算机视觉领域的众多应用中取得了突出的成果，如人脸识别、物体检测、医学影像分析等。由于其强大的特征学习能力，CNN 已成为图像分类任务中的主流模型之一。

1.3 主要研究工作及章节安排

本篇论文一共分为四章，下面将简要介绍每章的安排。

第一章为绪论，介绍了近年来深度学习的发展，以及 CNN 在图像分类中的

应用。

第二章为 CIFAR-10 的介绍，主要介绍了 CIFAR-10 的来源，已经它包含的图像的数量以及常见的用途。

第三章首先介绍了卷积神经网络的一些概念，接下来介绍了一个比较典型的模型 ResNet，接下来就是代码实现部分，采用了不同的训练轮次，不同的激活函数，以及不同的模型进行了分类精度的评估和分析。

最后为结论，总结了本文在研究过程中所做的内容和接下来可以更深入进行研究的方向。

第 2 章 数据集介绍

2.1 CIFAR-10

CIFAR-10（Canadian Institute for Advanced Research - 10）是由加拿大高级研究所于 2004 年创建的经典图像分类数据集。该数据集的产生旨在促进机器学习和计算机视觉领域的研究，为算法和模型提供一个标准化的测试基准。目前已经成为计算机视觉和机器学习社区中最常用的数据集之一。

研究人员通常使用 CIFAR-10 来开发和评估卷积神经网络（CNNs）和其他深度学习模型，因为它的规模适中，包含多个类别。它被作为评估图像分类算法能力的标准基准。

2.2 数据集组成

CIFAR-10 数据集包含 60,000 张 32x32 像素的彩色图像，分为 10 个不同的类别，每个类别有 6,000 张图像。这 10 个类别分别是：飞机（airplane）、汽车（automobile）、鸟类（bird）、猫（cat）、鹿（deer）、狗（dog）、青蛙（forg）、马（horse）、船（ship）、卡车（truck）。部分图片如图 2-1 所示：

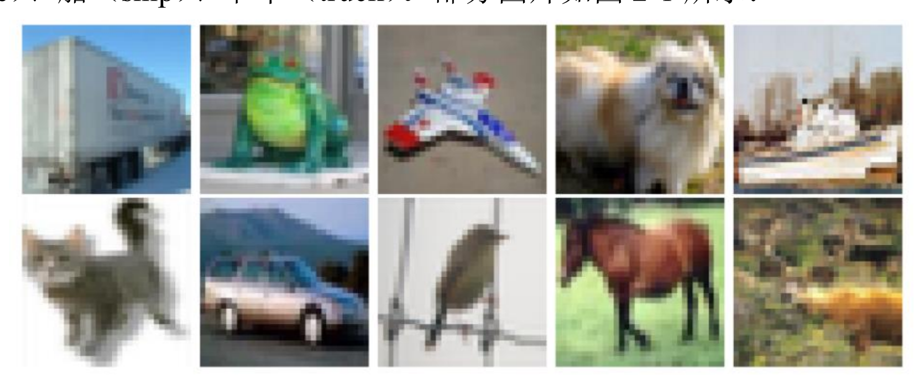


图 2-1 CIFAR-10 各种类图像展示

CIFAR-10 数据集被分为训练集和测试集，其中有 50,000 张图像用于训练，10,000 张用于测试。每个图像都带有一个标签，标明它属于这 10 个类别中的哪一个，因此适用于监督学习。

2.3 常见用途

CIFAR-10 主要用于评估和比较机器学习算法和模型在图像分类任务上的性能。研究人员和开发者经常使用这个数据集来测试新的图像分类算法，特别是

卷积神经网络（CNN）等深度学习模型。

同时，CIFAR-10 数据集为深度学习研究提供了一个有挑战性的任务，它的挑战在于图像分辨率相对较低，且图像中的对象可能出现在不同的位置和姿势，增加了分类的难度。研究人员利用 CIFAR-10 探索不同的神经网络架构、损失函数和优化方法，以提高模型对图像分类任务的性能。

此外，CIFAR-10 数据集也被广泛用于迁移学习的研究。由于其相对小的规模和多样性，研究人员可以使用 CIFAR-10 预训练的模型，然后将这些模型迁移到其它更大、更复杂的图像数据集上，从而提高模型的泛化能力。

总体而言，CIFAR-10 数据集在机器学习中发挥着重要作用，为研究人员和从业者提供了一个共同的基准，推动了图像分类领域的研究和发展。

第 3 章 基于 PyTorch 的图像分类研究

3.1 图像分类相关理论

3.1.1 卷积神经网络

卷积神经网络（CNN）是用来对如图像、视频等可以被看作多维网格的数据进行特征提取的网络结构，它具有稀疏连接、参数共享等特点，在计算机视觉领域内的检测、分类等各项任务中都取得优越的成果。以卷积层、池化层以及激活函数组合的卷积操作可以构建出一个完整的 CNN 网络，完成对输入数据的特征提取，再经由全连接层和目标函数完成指定的视觉任务。卷积神经网络参数更新方式采用反向传播算法将计算出的残差结果逐层向前传递，更新网络参数直至网络收敛。

（1）卷积层

卷积层是 CNN 的核心组件，通过卷积操作可以提取图像的局部特征，与传统神经网络中的矩阵乘法不同，卷积方式只需要通过卷积核对输入单元进行稀疏连接，局部交互。由于图像数据像素相邻部分的相关性更强，像素不相邻部分的相关性较弱，因此卷积的稀疏连接特性使其只需要关注像素邻域信息就可以有效提取特征，并且所需存储的模型参数更小，运算的效率也更高。卷积运算如图 3-1 所示：

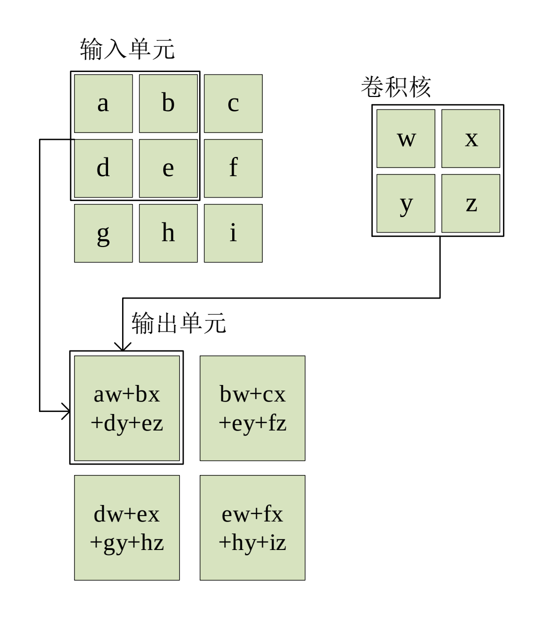


图 3-1 卷积运算过程

（2）池化层

经由卷积层操作后的特征再使用激活函数激活后,特征图像尺寸往往会非常大， 而池化层的作用则是调整输出特征图的尺度。池化操作使用特征图中某一位置的相邻特征的总体统计特征表示特征图中该位置的输出，这样可以使得特征图的通道数保持不变但尺度缩小，模型参数量也会相应变少，从而提高网络整体训练的效率，常见的池化方式有最大池化、平均池化等。

（3）全连接层

全连接层负责将卷积层提取的特征映射到最终的输出类别。在全连接层中，每个神经元与前一层的所有神经元相连接，通过学习权重和偏置来完成特征的组合和分类。

3.1.2 典型图像分类算法

Kaiming He 等人提出的 ResNet 深度残差网络是非常出名的卷积神经网络，它通过引入残差块和跳跃连接的设计，成功地缓解了梯度消失的问题。残差块允许信息直接传递到后续层，使得网络更容易学习到恒等映射。这种设计使得在训练深度网络时梯度能够更顺畅地传播，更容易优化。跳跃连接允许信息在网络中直接绕过中间层，有效地缓解了梯度的退化问题，使得网络可以更轻松地训练到数百层甚至上千层。这种结构的引入使得 ResNet 能够成功训练非常深的神经网络，取得了在图像分类等任务上的显著性能提升。网络核心的残差单元如图 3-2 所示：

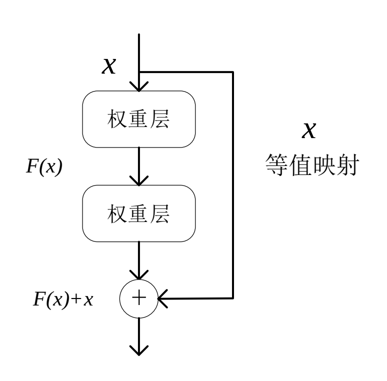


图 3-2 残差单元

3.1.3 激活函数

在深层网络中，激活函数的作用是为了实现网络的非线性特性。如果没有激活函数，无论网络的深度有多少，输入输出之间都可以使用线性组合来表示，这样就无法体现出深度网络与原始感知机之间的区别，因此需要使用激活函数来调

节深度网络。接下来将介绍常见的 Sigmoid 激活函数、ReLU 激活函数以及最近在图像任务中表现出色的 Swish 激活函数。

ReLU 激活函数是 G.Hinton 等人提出的，是至今仍在广泛使用的分段激活函数，其公式为：

$$ReLU(x)=max(0,x)$$

它的图像如下图 3-3 所示：

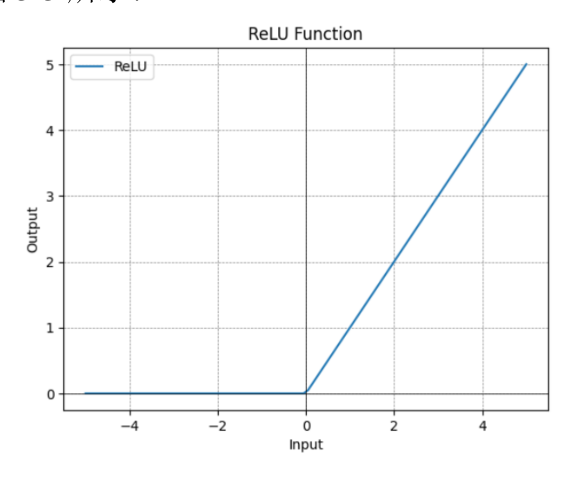


图 3-3 ReLU 函数图像

简言之，对于输入 x，如果 x 大于零，ReLU 输出 x；如果 x 小于或等于零，ReLU 输出零。

它能够缓解梯度消失的问题，同时运算速度非常快，仅需要确定输入的值与 0 的对比结果即可，收敛速度也比 Sigmoid 更快。但它也有缺点，在训练过程中，某些神经元可能永远不会被激活，称为“死亡神经元”。这可能导致梯度更新停滞，一种常见的解决方法是使用 Leaky ReLU 或其他变体。

总体而言，ReLU 是一种简单而有效的激活函数，特别在深度学习中取得了很好的效果。然而，在某些情况下，使用其他激活函数的变体可能更为合适，具体选择通常取决于任务的性质和网络结构的要求。

Leaky ReLU（泄漏线性整流单元）是 ReLU 的一种改进版本，旨在解决 ReLU 可能出现的“死亡神经元”问题。Leaky ReLU 允许负数部分有一个小的斜率，使得在负数输入时不是完全截断，从而提供了一些负数的输出。

Leaky ReLU 的数学表达式为：

$$Leaky\ ReLU(x)=max(\alpha x,x)$$

如下图 3-4 所示：

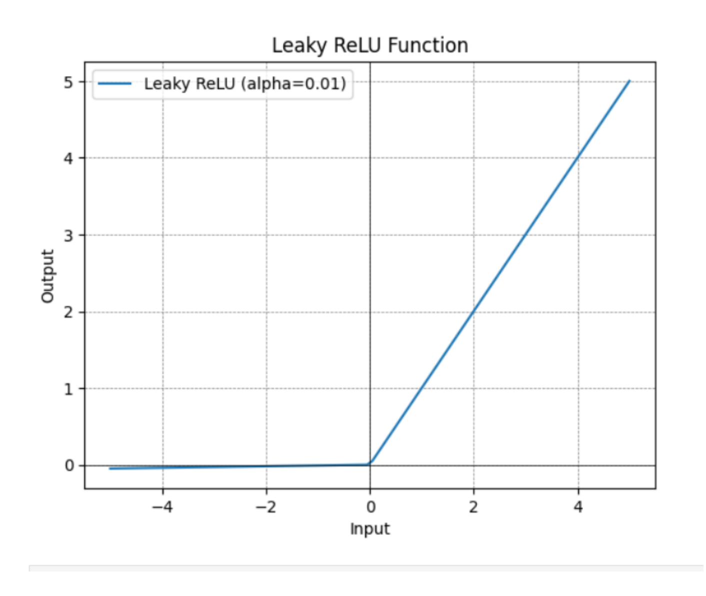


图 3-4 Leaky ReLU 函数图像

其中，α是一个小的正数，通常设置为接近零的较小值，如 0.01。这个小斜率在负数部分引入了一个非零的梯度，有助于在训练期间更好地传递梯度，防止某些神经元永远不被激活的问题，减轻了 ReLU 的一些问题。虽然 Leaky ReLU 减轻了“死亡神经元”问题，但仍然有可能一些神经元在训练中不被激活。在这种情况下，其他激活函数如 Parametric ReLU (PReLU)等可能是更好的选择。

在实际应用中，Leaky ReLU 经常用于深度学习模型的隐藏层，但具体的选择可能取决于数据集和任务的特性。

2017 年谷歌大脑提出了 Swish 激活函数，其公式为：

$$f(x) = x \cdot Sigmoid(x)$$

图像如图 3-5 所示:

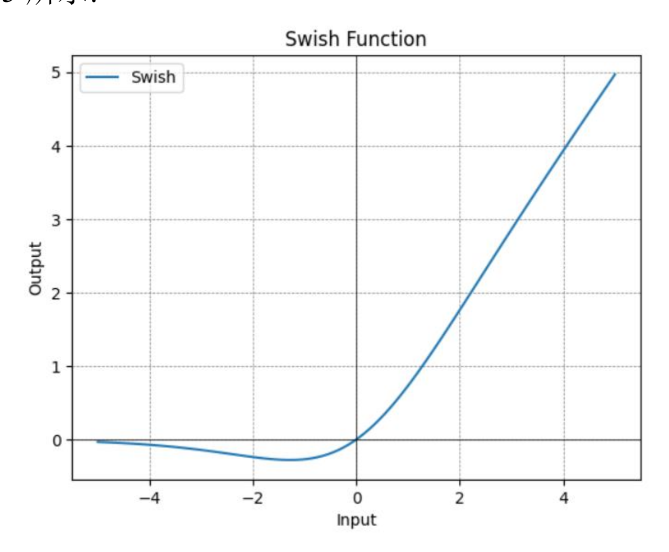


图 3-5 Swish 函数图像

这个激活函数非常简单只需要将输入乘以自己的 Sigmoid 值即可，它十分平滑，非单调且有下界无上界，在图像分类检测任务中与 ReLU 激活函数相比有着非常显著的性能提升。

3.2 图像分类代码实现

3.2.1 数据处理

我使用的数据集为 CIFAR-10，是从 https://www.cs.toronto.edu/~kriz/cifar.html 这个网站下载的。解压后可以看到文件的格式如图 3-6 所示：

batches.meta	2009/3/31 12:45	META 文件	1 KB
data_batch_1	2009/3/31 12:32	文件	30,309 KB
data_batch_2	2009/3/31 12:32	文件	30,308 KB
data_batch_3	2009/3/31 12:32	文件	30,309 KB
data_batch_4	2009/3/31 12:32	文件	30,309 KB
data_batch_5	2009/3/31 12:32	文件	30,309 KB
readme.html	2009/6/5 4:47	Microsoft Edge ...	1 KB
test_batch	2009/3/31 12:32	文件	30,309 KB

图 3-6 CIFAR-10 解压

可以通过官方给出的方法进行处理，同时通过进一步编写代码并将图片输出出来，同时输出图像所对应的标签。这部分我通过 cifar_10.ipynb 这部分代码来实现，最后得到的图片分为了测试集和训练集两个文件夹，分别命名为 test_cifar10、train_cifar10，对应的标签我分别存为了 testLables.csv、trainLables.csv，部分测试集的图片以及标签如图 3-7、3-8 所示：

	A	B	C	D	E	F	G	H
1	Image	Category						
2	1	cat						
3	2	ship						
4	3	ship						
5	4	airplane						
6	5	frog						
7	6	frog						
8	7	automobile						
9	8	frog						
10	9	cat						
11	10	automobile						
12	11	airplane						
13	12	truck						
14	13	dog						
15	14	horse						
16	15	truck						
17	16	ship						
18	17	dog						
19	18	horse						
20	19	ship						
21	20	frog						

图 3-7 测试集标签

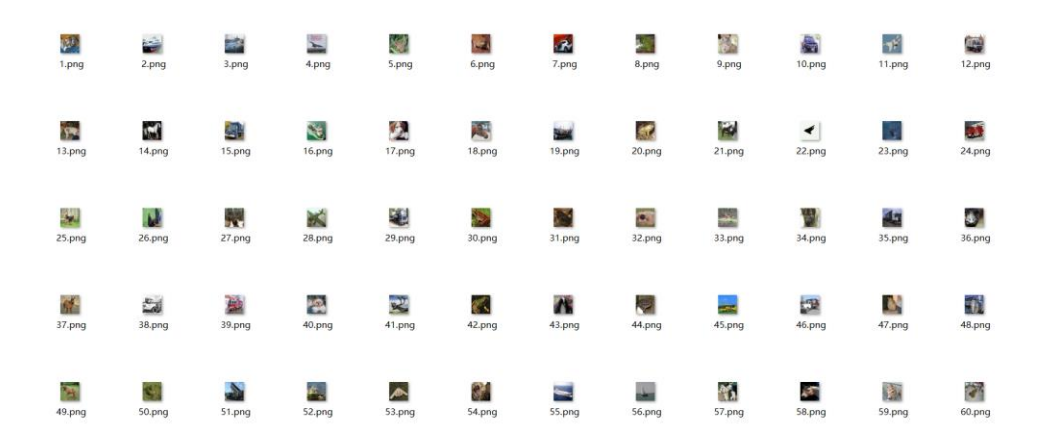


图 3-8 测试集图片

处理后的图片和标签就可以进行训练以及测试使用了。接下来将处理好的图片和标签放到对应的文件夹下，在这里，我放到了 imageclassification 中的 temp 文件夹下。

3.2.2 图像分类模型

我把代码放在了 ImageNet Classification.ipynb 这个文件中。这个代码的主要部分为模型部分，我的模型是一个卷积神经网络（CNN），输入数据经过第一个卷积层 (self.conv1)，然后通过 ReLU 激活函数，接着经过最大池化层 (self.pool) 进行降采样。然后，特征图经过第二个卷积层 (self.conv2)，再次通过 ReLU 激活函数，最后再经过一次最大池化层 (self.pool) 进行降采样。池化后的特征图通过 x.view(-1, 16*5*5) 操作被展平为一维向量。一维向量通过两个全连接层 (self.fc1 和 self.fc2)，每一层都应用 ReLU 激活函数。最终，通过输出层 (self.fc3) 得到模型的输出，对应于每个类别的得分。模型的结构如图 3-9 所示：

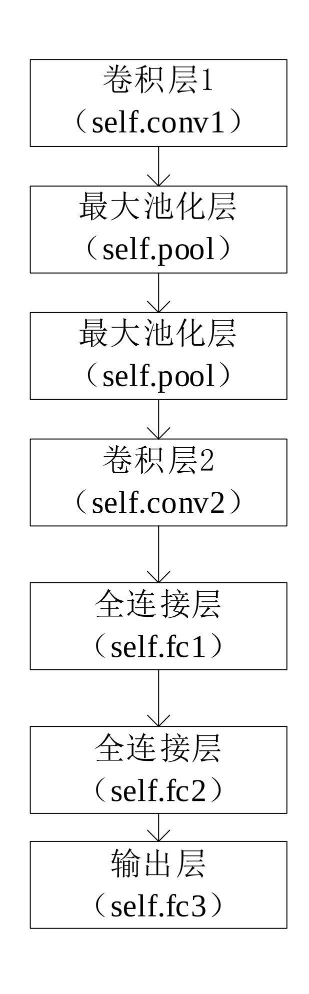


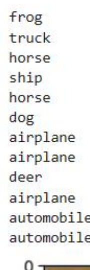
图 3-9 模型结构

3.2.3 训练以及评估

我把训练集中的 50000 张图片分为了 40000 张训练集和 10000 张验证集，利用验证集来评估训练的效果。

在训练中，通过遍历训练集中的小批量数据，通过前向传播计算模型的预测输出，然后使用交叉熵损失函数衡量预测值与真实标签之间的差异。接着进行反向传播，通过梯度下降优化器调整模型参数，以最小化损失。在每个小批量训练中，累积损失值并定期打印训练过程中的损失。最后，记录并输出最后一个小批量的训练损失。

我把所有的测试集进行了分批次预测，预测的结果保存到了 submission.csv 这个文件中，并通过代码比较 submission.csv 与 testLables.csv 中的标签是否相同，最后计算分类的准确度。这是训练了 60 轮的模型，使用随机的 12 个图像进行分类的结果，如图 3-6 所示：



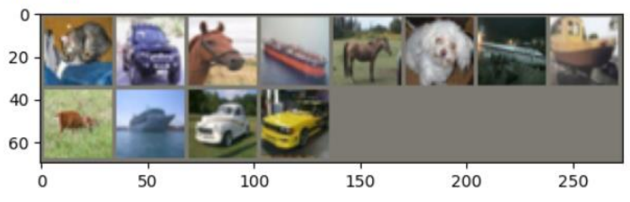


图 3-6 图像分类示例

从图中可以看出，这 12 张图片中，有 8 个图片的标签是正确的，分类的效果还是能够接受的。

在不同的训练次数下，准确度也在不断改变。不同训练的轮数对应的分类准确率如表 3-1 所示：

表 3-1 不同轮数对应分类精度表

训练轮数	分类精度
10	0.5744
30	0.6324
60	0.5765
100	0.5797

可以看到分类的精度在训练轮次为 30 的时候还是不错的，但是分类效果并不是随着训练轮次的增加而增加，可能在训练轮次增加后出现了过拟合，这个需要在后面的模型改进中多加关注。

3.2.3 激活函数改进

接下来我使用了不同的激活函数进行了分类精度评估，我接下来使用了 Leaky ReLU、Swish 这两个激活函数来来进行实验，其中 Leaky ReLU 的α设置为了 0.01。采用 Leaky ReLU、Swish 激活函数的代码我分别保存到了 ImageNet Classification_Leak.ipynb、ImageNet Classification_Swish.ipynb 这个文件中，运行后分别得到了不同训练轮数下的分类精度。

表 3-2 分类精度表

训练轮数	Leaky ReLU	Swish
10	0.61	0.5787
30	0.6292	0.6193
60	0.5917	0.5872
100	0.5915	0.5985

从表 3-2 中可以看出，Leaky ReLU 在较少的训练轮次下分类效果还是很不错的，在 10 轮次的训练下就能达到 0.61 的分类精度。同时使用 Leaky ReLU 这个激活函数后，整体的分类精度相对于 ReLU 有一定的提升。同时在这些激活函数中仍然存在随着轮次的增加，分类效果反而下降的情况，所以问题应该出现在模型部分，在接下来的模型改进中我将会采取措施来改进。

3.2.4 模型改进

由于前面出现了随着训练轮次增加，而分类效果下降的问题，我通过进一步调整模型，增加了 Dropout 层，以 0.5 的概率进行神经元的随机失活，有助于防止过拟合。调增后的代码我放到了 Improve.ipynb 中，模型代码如图 3-7 所示：

```
import torch.nn as nn
import torch.nn.functional as F

class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.conv1 = nn.Conv2d(3, 32, 3, padding=1)
        self.conv2 = nn.Conv2d(32, 64, 3, padding=1)
        self.conv3 = nn.Conv2d(64, 128, 3, padding=1)
        self.pool = nn.MaxPool2d(2, 2)
        self.fc1 = nn.Linear(128 * 4 * 4, 512)
        self.fc2 = nn.Linear(512, 256)
        self.fc3 = nn.Linear(256, 10)
        self.dropout = nn.Dropout(0.5)

    def forward(self, x):
        x = self.pool(F.relu(self.conv1(x)))
        x = self.pool(F.relu(self.conv2(x)))
        x = self.pool(F.relu(self.conv3(x)))
        x = x.view(-1, 128 * 4 * 4)
        x = F.relu(self.fc1(x))
        x = self.dropout(x)
        x = F.relu(self.fc2(x))
        x = self.fc3(x)
        return x

# 创建改进后的模型
net = Net()
```

图 3-7 调增后模型

卷积层 1 接收输入图像，通过 3x3 的卷积核提取 32 个特征图。通过 ReLU 激活函数后，特征图被传递给池化层。

卷积层 2 接收上一层的特征图，通过 3x3 的卷积核提取 64 个特征图。通过 ReLU 激活函数后，特征图被传递给池化层。

卷积层 3 接收上一层的特征图，通过 3x3 的卷积核提取 128 个特征图。通过

ReLU 激活函数后，特征图被传递给池化层。

全连接层 1 将经过卷积层和池化层的特征图展平成一维向量，然后通过全连接层，将其映射到一个 512 维的特征空间。这一步的激活函数是 ReLU。

全连接层 2 输入为 512 维的特征，输出为 256 维，同样使用 ReLU 激活函数。

全连接层 3 输入为 256 维的特征，输出为 10 维，对应于分类任务的类别数量。

最大的改动是增加了 Dropout 层，位于全连接层之间，以 0.5 的概率对神经元进行随机失活。这有助于防止过拟合。

最后的输出层是一个全连接层，输出为 10 维，对应于图像分类任务中的 10 个类别。在这里不使用激活函数，因为这是一个多类别分类问题，通常在损失函数中集成了 softmax 激活。

这个模型通过卷积层和池化层提取图像的特征，然后通过全连接层进行分类。Dropout 层有助于防止过拟合。整个过程通过前向传播完成，其中每个层的输出都是下一层的输入。

使用改动后的模型进行分类，可以看到分类结果如表 3-3 所示：

表 3-3 不同模型分类精度表

训练轮数	原模型（ReLU）分类精度	改进后模型分类精度
10	0.5744	0.6856
30	0.6324	0.7344
60	0.5765	0.7369
100	0.5797	0.7509

可以看到，改进后的模型分类精度的得到了明显的提高，在 100 轮次的训练下，分类精度可以达到 0.7509，同时过拟合问题也得到了解决，由于设备限制，这里不再继续增加训练轮次，但是分类精度有随着训练批次继续增加的趋势。

3.2.5 ResNet 模型

接下来我使用了 ResNet 模型进行分类效果评估，ResNet 模型的代码放置在了 ResNet.ipynb 中并进行训练，得到的分类精度表如表 3-4 所示：

表 3-4 ResNet 模型分类精度表

训练轮数	原模型（ReLU）分类精度	ResNet 模型分类精度
10	0.5744	0.7088

由于设备的限制，我仅仅使用了 10 轮次进行训练，可以看到最后得到的分类效果相对于原模型还是有很大的提升，达到了 0.7088。同时相对于改进过的模型 Improve 也是提升了 0.0232。

3.3 章节小节

在以上的优化实验中，我不断改进模型，使用不同的激活函数和训练轮次来提升图像的分类效果，在整个实验中，我得到的分类效果最好的模型为 Improve 这个模型训练了 100 次，得到的准确率为 0.7509，同时这个模型的分类效果随着训练轮次的增加还在不断提升，但是由于设备的限制，我没有继续增加训练轮次。后面如果有更好的设备，可以继续使用这个模型进行改进，同时可以利用 GPU 的高速的计算性能来减少训练所需要的时间。

本文中使用的所有代码，都放到了 https://github.com/LAS666/DeepLearingCode.git 这个仓库。

结 论

在本文中，我最开始使用了一个简单的卷积神经网络构建的模型，但是它的效果并不理想，同时在训练轮次增大的时候会产生过拟合现象，我通过改变它的激活函数，发现仍然没有解决这个问题，在接下来，我修改了自己使用的模型，增加了 Dropout 层，并改变了模型的结构，接下来增加训练轮次分类效果逐步提高，在一定程度上解决了过拟合问题，同时优化后的模型的分类效果比原始模型要好很多，在 100 轮次的训练下可以达到 0.7509。这个分类效果还是很不错的。

通过上述的一系列实验已经模型的优化，我对于基于深度学习的图像分类更加的了解了，同时我也看到了它的前景，未来一定会涌现出更多的架构和模型，分类效果也将不断提高，我也将继续对于这个方向的研究。